deterministic logic and arithmetic instructions by:

ensuing processing of Instruction #1 on the first cycle in which an OpCode, source address and destination address are read from an Instruction Memory;

providing a source address to the interface blocks for Instruction #1 on the second cycle, and ensuing processing of Instruction #2 in which the OpCode, source address and destination address are read from the Instruction Memory;

A1

when data for Instruction #1 is available from the interface blocks on the third cycle, providing the source address to the interface blocks for Instruction #2, and ensuing processing of Instruction #3 in which the OpCode, source address and destination address are read from the Instruction Memory;

processing data from the interface blocks for Instruction #1 on the fourth cycle, and when data for Instruction #2 is available from the interface blocks, providing the source address to the interface blocks for Instruction #3 and ensuing processing of Instruction #4 in which the OpCode, source address and destination address are read from the Instruction Memory;

providing destination and write controls of Instruction #1 for the interface blocks on the fifth cycle, processing data from the interface blocks for Instruction #2, and when data for Instruction #3 is available from the interface blocks, providing the source address to the interface blocks for Instruction #4 and ensuing processing of Instruction #5 in which the OpCode, source address and destination address are read from the Instruction Memory; and

when Instruction #1 is retired on the sixth cycle, providing destination and write controls of Instruction #2 for the interface blocks, processing the data from the interface blocks for Instruction #3, and when data for Instruction #4 is available from the interface blocks, providing the source address to the interface blocks for the instruction #5 and ensuing processing of Instruction #6 in which the OpCode, source address and destination address are read from the Instruction Memory.

A1

22. The host-fabric adapter as claimed in claim 21, wherein said Micro-Engine (ME) is configured to ensure that data dependency between contiguous Micro-instructions is dealt correctly.

23. The host-fabric adapter as claimed in claim 1, wherein said Micro-Engine (ME) processes multiple ME instructions in parallel, when said ME instructions are non-deterministic instructions by:

ensuing processing of Instruction #1 on the first cycle in which an OpCode, source address and destination address are read from an Instruction Memory;

providing the source address to the interface blocks for Instruction #1 on the second cycle, and ensuing processing of Instruction #2 in which the OpCode, source address and destination address are read from the Instruction Memory;

when data for Instruction #1 is available from the interface blocks on the third cycle, and a conditional Jump instruction based on Flags is set for Instruction #2, ensuing

processing of Instruction #3 in which the OpCode, source address and destination address are read from the Instruction Memory;

processing data from the interface blocks for Instruction #1 on the fourth cycle, providing the source address to the interface blocks for Instruction #3, ensuing processing of Instruction #4 in which the OpCode, source address and destination address are read from the Instruction Memory;

when data for Instruction #3 is available from the interface blocks on the fifth cycle

A1

if the Jump condition is not TRUE, ensuing processing of Instruction #5 in which the OpCode, source address and destination address are read from the Instruction Memory; and

if the Jump condition is TRUE, ensuing processing of the conditional Jump instruction in which the OpCode, source address and destination address are read from the Instruction Memory corresponding to a Jump Address, and when Instruction #1 is retired on the sixth cycle, flushing Instruction #3, providing the source address to the interface blocks for the conditional Jump instruction corresponding to the Jump Address.

24. The host-fabric adapter as claimed in claim 23, wherein said Micro-Engine (ME) is configured to ensure that only latest data from the interface blocks is used and correct data is written to the interface blocks.

25. The host-fabric adapter as claimed in claim 1, wherein said Micro-Engine (ME) processes multiple tasks in parallel by:

ensuing processing of Instruction #1 on the first cycle in which an OpCode, source address and destination address are read from an Instruction Memory;

providing the source address to the interface blocks for Instruction #1 on the second cycle, and ensuing processing of Instruction #2 indicating a Task Switching Instruction in which the OpCode, source address and destination address are read from the Instruction Memory;

A1

when data for Instruction #1 is available from the interface blocks and there is no data processing on the third cycle for Instruction # 2, ensuing processing of Instruction #3 for a new task in which the OpCode, source address and destination address are read from the Instruction Memory;

processing data for Instruction #1 from the interface blocks on the fourth cycle and providing the source address to the interface blocks for Instruction #3 for the new task;

providing destination and write controls of Instruction #1 for the interface blocks on the fifth cycle for the old task and, when data for the new task for Instruction #3 is available from the interface blocks, providing the source address to the interface blocks for Instruction #4 for a new task and ensuing processing of Instruction #5 for the new task in which the OpCode, source address and destination address are read from the Instruction Memory;

when Instruction #1 is retired on the sixth cycle, processing data from the interface blocks for Instruction #3 for the new task, and when data for Instruction #4 is available from the interface blocks for the new task, providing the source address to the interface blocks for Instruction #5 for a new task and ensuing processing of Instruction #6 for the new task in

which the OpCode, source address and destination address are read from the Instruction Memory; and

A1

when Instruction #2 is retired on the seventh cycle, providing destination and write controls for the interface blocks for Instruction #3 for the new task, processing data from the interface blocks for Instruction #4 for the new task, and when data for Instruction #5 is available from the interface blocks for the new task, providing the source address to the interface blocks for Instruction #6 for a new task and ensuing processing of Instruction #7 for the new task in which the OpCode, source address and destination address are read from the Instruction Memory.

26. The host-fabric adapter as claimed in claim 1, wherein said Micro-Engine (ME) is implemented to achieve a throughput of one instruction per clock for logic and arithmetic instructions by processing multiple instructions in parallel in multiple pipelines.

27. The host-fabric adapter as claimed in claim 1, wherein said Micro-Engine (ME) is implemented to achieve a throughput of one instruction per clock for logic and arithmetic instructions even in the event of data-dependency between contiguous instructions

28. The host-fabric adapter as claimed in claim 1, wherein said Micro-Engine (ME) is implemented to handle multiple instructions at any given time even in the event of uncertainty of the next instruction to be executed

29. The host-fabric adapter as claimed in claim 1, wherein said Micro-Engine (ME) is implemented to achieve a throughput of one instruction per clock even in the case of non-determinism of the next instruction to be executed.

30. The host-fabric adapter as claimed in claim 1, wherein said Micro-Engine (ME) is implemented to perform multi-tasking (multi-threading) with minimal hardware and graceful integration into normal processing.

A1

31. The host-fabric adapter as claimed in claim 1, wherein said Micro-Engine (ME) is implemented to perform multi-tasking (multi-threading) with non-duplication or non-dedication of hardware computing resources per task.

32. A host system, comprising:

a host processor;

a host memory;

a host fabric adapter connected to said host processor and said host memory via a system bus, and installed to access to a switched fabric, said host fabric adapter comprising:

a host interface arranged to interface said system bus;

a serial interface arranged to receive and transmit data from said switched fabric;

at least one Micro-Engine (ME) arranged to establish connections and support data transfers, via a switched fabric, in response to work requests for data transfers;

interface blocks arranged to interface said switched fabric and said system bus, and send/receive work requests and/or data for data transfers, via said switched fabric, and configured to provide context information needed for said Micro-Engine (ME) to process said work requests for data transfers, via said switched fabric,

wherein said Micro-Engine (ME) is implemented with a pipelined instruction execution architecture to handle one or more ME instructions and/or one or more tasks so as to process data for data transfers.

A1

33. The host system as claimed in claim 32, wherein said Micro-Engine (ME) is implemented to achieve a throughput of one instruction per clock for logic and arithmetic instructions by processing multiple instructions in parallel in multiple pipelines.

34. The host system as claimed in claim 32, wherein said Micro-Engine (ME) is implemented to achieve a throughput of one instruction per clock for logic and arithmetic instructions even in the event of data-dependency between contiguous instructions

35. The host system as claimed in claim 32, wherein said Micro-Engine (ME) is implemented to handle multiple instructions at any given time even in the event of uncertainty of the next instruction to be executed

36. The host system as claimed in claim 32, wherein said Micro-Engine (ME) is implemented to achieve a throughput of one instruction per clock even in the case of non-determinism of the next instruction to be executed.

37. The host system as claimed in claim 32, wherein said Micro-Engine (ME) is implemented to perform multi-tasking (multi-threading) with minimal hardware and graceful integration into normal processing.

A1

38. The host system as claimed in claim 32, wherein said Micro-Engine (ME) is implemented to perform multi-tasking (multi-threading) with non-duplication or non-dedication of hardware computing resources per task.

39. The host system as claimed in claim 32, wherein said Micro-Engine (ME) processes multiple ME instructions in parallel, when said ME instructions are deterministic logic and arithmetic instructions by:

ensuing processing of Instruction #1 on the first cycle in which an OpCode, source address and destination address are read from an Instruction Memory;

providing a source address to the interface blocks for Instruction #1 on the second cycle, and ensuing processing of Instruction #2 in which the OpCode, source address and destination address are read from the Instruction Memory;

when data for Instruction #1 is available from the interface blocks on the third cycle, providing the source address to the interface blocks for Instruction #2, and ensuing processing of Instruction #3 in which the OpCode, source address and destination address are read from the Instruction Memory;

processing data from the interface blocks for Instruction #1 on the fourth cycle, and when data for Instruction #2 is available from the interface blocks, providing the source address to the interface blocks for Instruction #3 and ensuing processing of Instruction #4 in which the OpCode, source address and destination address are read from the Instruction Memory;

A1

providing destination and write controls of Instruction #1 for the interface blocks on the fifth cycle, processing data from the interface blocks for Instruction #2, and when data for Instruction #3 is available from the interface blocks, providing the source address to the interface blocks for Instruction #4 and ensuing processing of Instruction #5 in which the OpCode, source address and destination address are read from the Instruction Memory; and

when Instruction #1 is retired on the sixth cycle, providing destination and write controls of Instruction #2 for the interface blocks, processing the data from the interface blocks for Instruction #3, and when data for Instruction #4 is available from the interface blocks, providing the source address to the interface blocks for the instruction #5 and ensuing processing of Instruction #6 in which the OpCode, source address and destination address are read from the Instruction Memory.

40. The host system as claimed in claim 39, wherein said Micro-Engine (ME) is configured to ensure that data dependency between contiguous Micro-instructions is dealt correctly.

41. The host system as claimed in claim 32, wherein said Micro-Engine (ME) processes multiple ME instructions in parallel, when said ME instructions are non-deterministic instructions by:

A1

ensuing processing of Instruction #1 on the first cycle in which an OpCode, source address and destination address are read from an Instruction Memory;

providing the source address to the interface blocks for Instruction #1 on the second cycle, and ensuing processing of Instruction #2 in which the OpCode, source address and destination address are read from the Instruction Memory;

when data for Instruction #1 is available from the interface blocks on the third cycle, and a conditional Jump instruction based on Flags is set for Instruction #2, ensuing processing of Instruction #3 in which the OpCode, source address and destination address are read from the Instruction Memory;

processing data from the interface blocks for Instruction #1 on the fourth cycle, providing the source address to the interface blocks for Instruction #3, ensuing processing of Instruction #4 in which the OpCode, source address and destination address are read from the Instruction Memory;

when data for Instruction #3 is available from the interface blocks on the fifth cycle

if the Jump condition is not TRUE, ensuing processing of Instruction #5 in which the OpCode, source address and destination address are read from the Instruction Memory;

if the Jump condition is TRUE, ensuing processing of the conditional Jump instruction in which the OpCode, source address and destination address are read from the Instruction Memory corresponding to a Jump Address, and when Instruction #1 is retired on the sixth cycle, flushing Instruction #3, providing the source address to the interface blocks for the conditional Jump instruction corresponding to the Jump Address.

A1

42. The host system as claimed in claim 41, wherein said Micro-Engine (ME) is configured to ensure that only latest data from the interface blocks is used and correct data is written to the interface blocks.

43. The host system as claimed in claim 32, wherein said Micro-Engine (ME) processes multiple tasks in parallel by:

ensuing processing of Instruction #1 on the first cycle in which an OpCode, source address and destination address are read from an Instruction Memory;

providing the source address to the interface blocks for Instruction #1 on the second cycle, and ensuing processing of Instruction #2 indicating a Task Switching Instruction in which the OpCode, source address and destination address are read from the Instruction Memory;

when data for Instruction #1 is available from the interface blocks and there is no data processing on the third cycle for Instruction # 2, ensuing processing of Instruction #3 for a new task in which the OpCode, source address and destination address are read from the Instruction Memory;

processing data for Instruction #1 from the interface blocks on the fourth cycle and providing the source address to the interface blocks for Instruction #3 for the new task;

A1

providing destination and write controls of Instruction #1 for the interface blocks on the fifth cycle for the old task and, when data for the new task for Instruction #3 is available from the interface blocks, providing the source address to the interface blocks for Instruction #4 for a new task and ensuing processing of Instruction #5 for the new task in which the OpCode, source address and destination address are read from the Instruction Memory;

when Instruction #1 is retired on the sixth cycle, processing data from the interface blocks for Instruction #3 for the new task, and when data for Instruction #4 is available from the interface blocks for the new task, providing the source address to the interface blocks for Instruction #5 for the new task and ensuing processing of Instruction #6 for the new task in which the OpCode, source address and destination address are read from the Instruction Memory; and

when Instruction #2 is retired on the seventh cycle, providing destination and write controls for the interface blocks for Instruction #3 for the new task, processing data from the interface blocks for Instruction #4 for the new task, and when data for Instruction #5 is available from the interface blocks for the new task, providing the source address to the

A1 concl.

interface blocks for Instruction #6 for the new task and ensuing processing of Instruction #7 for the new task in which the OpCode, source address and destination address are read from the Instruction Memory.